

HUGE V2.0

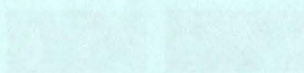


The image processor

Reference Manual

Copyright 1991 by UGA software

HUGE V.S.O.



The World's Largest

International

THE WORLD'S LARGEST

Table of contents

Table of contents	1
Introduction	3
Objects	4
Images	4
Sprites	4
Brushes	5
Bobs.....	5
Icons	5
Palettes	5
Setup.....	6
Terminology.....	7
Drawing area	7
Current position.....	7
Idle state.....	7
Drawing	7
Actions	7
Operations.....	7
Wait state.....	7
Beep.....	8
Creating images.....	9
Zoom	9
Cut	9
Paste	9
Fill	10
Mirrors & rotators.....	10
Font selector.....	10
Write text	10
Palette.....	10
Undo	10
Workbench.....	10
Page #.....	10
Operability.....	11
Keyboard equivalents.....	11
Zoom concept	11
Input/Output.....	13
Description of the Huge I/O system	13
Using requesters	15
Messages	15
Simple input	15
Icon tool types.....	15
Zoom resolution.....	15
Screen mode	15
Palette.....	16
Font selector	16
Loader	16
Saver	17

Using menus.....	18
Project menu.....	18
Preferences menu	18
Icon menu	18
Special menu.....	19
Requirements.....	20
Memory conservation.....	21
Disk based libraries.....	21
Raster memory	21
I/O and clip rastport.....	21
Startup options.....	23
CLI start.....	23
Workbench start.....	23
Bibliography.....	25
A final word.....	26

Introduction

Amiga led the way in a new concept of user interface, the communication between machine and man. That is easier for the user, harder for the programmer. **Huge** is basically a programmer's utility, easing the pain of the development of this interface. It practically eliminates the visible aspect, saving time for concentrating on the more important part, the program kernel. No matter how powerful a program might be, the lack of a fine working environment is a disaster. The program looks aren't enough to make it great, but there is no great program without fine looks. **Huge** provides with minimal effort, every imagery an Amiga programmer encounters: *Images* for gadgets, menus, animation and logos, *sprites* for pointers, *palettes* for filling the colour registers, even the *icon* for the final program. And all this in a fine and functional environment.

The casual Amiga user is by no means left out. Beginners in programming will benefit from the *Bob* image format which can be used in their Basic steps. Everybody will be, sooner or later, tempted to modify an icon, or provide an alternate image for highlighting, all these are a breeze using **Huge's** built-in *Icon editor*. Finally, the popular *Brush* format is available, mainly as an import facility, Amiga has much more powerful programs for creating visual masterpieces. Indeed, all the visual Amiga formats are dealt with.

Huge is able to differentiate between PAL & NTSC Amigas. It takes advantage of PAL's excess number of display lines to create greater images, so Europeans will have higher imagery than the Americans.

Objects

Huge is an image processor. It can load, modify and save back the most frequently used Amiga image formats, converting them if desired to any other format. *Sprites* and *Icons* are always displayed on a fixed 4 colour non-interlaced screen, which is high resolution for Icons. There is no sense in drawing them in a different display, else there will be no match between what you draw and what you see when you use them. Therefore, Huge makes sure that these objects receive the appropriate display. The rest objects put no restrictions on the screen mode. Following is a description of all objects.

Images

Programmers stumble on them all the time. If you are a programmer, you know what I mean. Every visible Intuition item has its corresponding Image, ie. a descriptive structure and the imagery itself. Huge will take the imagery user has drawn and output to disk the numbers that make up the display (an array of UWORDS) and the descriptive structure (struct Image) in ASCII, bitplane by bitplane, in a format that every C compiler or assembler will understand. The output language is selected by the corresponding Preferences menu (*Language*). The numbers themselves are same, the difference is in the language format.

Huge allows for different file names, but the contained source code always gives the same names for the UWORD array & Image struct, the programmer might need to rename these variables before compilation. Assembly programmers must put the image data in a chip section.

Huge will load any source code Image it has produced, expecting to find everything exactly the way it saved it. The comments it places in the beginning are vital for identification and object size and display. Each line must be as saved, ie. same number of hexadecimal numbers on line, left indented by a tab, not 8 spaces.

compatibility note: Image format has undergone some minor alterations since the first version of this editor. The changes concern the initial comments. In order to load an Image produced by Huge v1.0 or v1.1, draw a scratch with v2.0, using the same display (depth & resolution), and save it. DEPTH, HIRES and LACE comments of the old Image must be replaced by the DISPLAY comment of the freshly created one.

Sprites

Mainly used as pointers. The output is source code, like Images, either in C or Assembly format, an array of UWORDS containing the numbers that make up the sprite imagery. This array can be directly fed to a *SetPointer()* call, it includes 4 more UWORDS (2 in the beginning and 2 at the end), complying to Intuition peculiarities. Unlike Images, Sprites must be less than 16 pixels wide, and are always displayed in full low resolution.

The colours 1, 2 and 3 which are used to draw them in Huge must be placed in colour registers 17, 18, and 19 respectively, and colour 0 is transparent. In the preferences tool, the sprite editor has the transparent colour in the fourth position, but in fact it is there colour 0 of the screen it opens in. This erroneously leads guys (including myself) to assume that the fourth colour is transparent. Now the matter is cleared once and for all. User must specify the pointer hot spot, depending on the sprite form.

compatibility note: see Image note.

All the rest objects are standard Amiga formats.

Brushes

The most frequently used Amiga image format (IFF/ILBM). All paint packages output in this format (eg. Deluxe Paint). It is not particularly useful for anything, except for a standard way for paint programs to communicate with each other. The reason it is one among *Huge's* objects, is to create an imagery using a heavy paint program, and transform it to a 'useful' object, like Image, Sprite, Bob or Icon. *Huge* won't load HAM brushes (ie. DigiPaint is off, no matter how much I love it). *Huge* is limited to a maximum of five bitplanes (32 colours in low resolution), and the sixth bitplane of HAM brushes is too important to be neglected.

Bobs

Bobs are the images used in AmigaBASIC programs. Before *Huge*, Basic programmers were limited to four colour bobs, created by the primitive ObjEdit utility supplied with AmigaBASIC. *Huge* enables creation of bobs up to 32 colours deep, in any screen resolution. *Huge* Bobs are *bobs*, not *sprites*, which may be a bit slower than sprites, but can be as big and colourful as desired. Anyway sprites are limited to four colours and ObjEdit is quite fine. Like ObjEdit, the flags set are SAVEBACK and OVERLAY, which means save background before rendering and treat colour 0 as transparent. Moreover, no Collision or ImageShadow plane is saved, like ObjEdit. In order to understand the meaning of these planes, a Basic programmer will have to read the ROM Kernel manuals, and a so daring guy won't be using Basic at all. For further information on bobs, users should reference the AmigaBASIC manual.

Note that bobs are a primitive image format, having no identification embedded in their file. Thus it is possible to load an alien file as a bob, if by chance some values in its beginning seem to be valid bob characteristics. You surely won't like the imagery, though.

Icons

Everybody has seen and heard of icons, used by the WorkBench user interface to represent graphically the various parts of AmigaDOS file system. Despite their wide use, there is limited indepth documentation on the standard manuals supplied with every Amiga. Icons are quite complex, and should not be taken lightly; messing around with unknown icon characteristics can do more harm than good. For a complete icon reference, users must refer to the Amiga ROM Kernel reference manual (Includes & Autodocs under the Workbench chapter). See also the discussion under the *Icon menu*.

Palettes

Palettes are not really objects, there is no image information in them, still they are extremely important, since they contained the RGB values that make up the colours used in drawing. Like Images and Sprites, palettes are source code, an array of UWORDS that can be fed to a *LoadRGB4()* call, to give your custom screen the colours you've selected. All 32 colour registers are saved, the output language depends on the current object. If it is Image or Sprite, the language used is the language currently selected, else the language used is C. Since Images and Sprites are saved with no colour information, it is vital to have the colour registers. For the rest objects, this palette I/O mechanism provides an easy way of accessing various palettes.

Notice once more that *Sprites* have an peculiar palette use. Before using a palette you saved (after drawing a sprite), you must move the colour entries 1, 2 & 3 (index starts at 0) to positions 17, 18 & 19, otherwise the sprite will be seen in different colours than those used to draw it. It is guarantied that you won't be able to recognise it.

compatibility note: In order to load a palette saved with an older version of *Huge*, again notice the new comments and place them in the old palette, using any text editor.

Setup

When the user starts Huge, either from Workbench or CLI, two screens open, one for the gadgets and menus, the other for drawing. Each screen has its own pointer and is automatically selected by positioning the mouse pointer anywhere over the screen.

The gadget screen is separate, so the imagery can survive any user configuration of colours and resolution. There are three groups of gadgets, two for various operations, one for the drawing tool used. Four menus along with a coordinates display window complete the display. The drawing screen consists of a bar holding some colour selection gadgets, the zoom window and the actual image window.

If Huge cannot find a preferences file, it defaults editing a C-language image, in a HIRES non-interlaced screen with 8 default colours and a default zoom resolution. The type of object being edited has nothing to do with the actual drawing, it only affects the methods of input and output to disks. The primary selections the user should make, are screen mode, palette, and object type. These preferences can be saved, so that they don't have to be changed every time the program is run.

compatibility note: preferences files saved with Huge v1.0 or v1.1 cannot be used in the new version of this editor.

Terminology

Before proceeding, some frequently used terms are explained.

Drawing area

The draw screen portion available for drawing. It is roughly the 3/5 of available width, and full height. A portion of the drawing area is displayed magnified on the zoom window.

Current position

Current position is always an offset in the drawing area. If the pointer is in the zoom window, current position is assumed the one corresponding to the spot of the actual image being magnified. Finally if the pointer is outside drawing area and zoom window, current position is assumed either 0 or max-position, depending on the pointer position.

Idle state

The state when nothing happens. No operation gadget or menu is selected and both mouse buttons are up.

Drawing

If while in idle state the left mouse button is pressed, drawing is initiated. It is affected by the current position and the drawing tool currently selected. Some tools use intermediate drawing in complement mode, till the desired form is created. Drawing is terminated by releasing the left mouse button. The right mouse button signals end of line in the connected lines tool.

Actions

Three elementary operations: *get_point*, *get_position* & *get_size*. Launched by operations needing a single point, or a rectangle positioning or sizing. *Get_point* uses current position to provide a spot in the drawing area. *Get_position*, positions a rectangle on the drawing area, dragging it from the top left corner (current position). *Get_size* is always a second stage action of a cut procedure, following a *get_point* action which provided a vertex of the rectangle. *Get_size* sizes the rectangle in any direction inside the drawing area by moving the vertex opposite to the one provided by the preceding *get_point*. All actions use temporary lines which are erased when terminated. Actions are normally terminated by left mouse button clicks, but also by aborting the operation that launched them. A *get_point* action followed by *get_size* is a *cut procedure*.

Operations

There are two kind of operations, those that use actions and those that have an immediate effect. The first type is initiated by selecting a gadget from the first group (TOGGLESELECT gadgets), while in idle state. The gadget remains selected and one or more actions are launched. When all actions are finished, the operation acts on the screen portion provided by the action, and the gadget is deselected. This type of operation can be aborted by selecting the same or another operation of the same type. Some operations (fill and paste) autorepeat until they are manually aborted. The second type of operations don't need a screen portion to act on, so they are initiated immediately after the (non-TOGGLESELECT) gadget selection (second group of gadgets).

Wait state

When Huge is performing a time consuming operation, or a requester or window opens in the control screen, the normal communication between Huge and the user is suspended; no gadgets or menu items can be selected, no drawing can happen, and the

pointer turns to a sleep pointer. Normal operation is restored as soon as the time consuming operation finishes, or the user responded to the active requester or window.

Beep

Both screens flash quickly. Something went wrong, but it wasn't serious enough to launch a message.

Creating images

The primary decisions the user should make before drawing is screen resolution and depth, which is exactly the same display that he will use in his/her program. A ~~three plane~~ (8 colours) screen is a good trade off between memory conservation and looks. The palette selection is also very important, not only the colours themselves, but also their relative position in the palette. If you plan to use complement highlighting for your gadgets, then colours of inverse intensity should be in complement positions, providing a 3-D effect.

Once the screen resolution and colours are selected, we are ready to begin. Various drawing tools are available: points, lines, boxes, circles, ellipses and connected lines. Just select a gadget among the third gadget group and you got the corresponding tool. Note that circle radius is determined only by the vertical distance from the centre.

The colour used can be altered by hitting a colour gadget. The colour currently selected is displayed in the right end of the colour bar. The deeper the screen, the more colours you get. For screens with few colours, a primitive dithering technique can be used (in fact checkered). Setting *Pattern active*, and selecting *Background colour* from the special menu, gives this ability. This mixed colour is used in every line, point or fill operation, until another colour combination is selected, or the pattern mode is made inactive. A little bug here, circles and ellipses don't comply to pattern, they just use foreground colour instead. Well, that's Amiga's bug, not Hugs's. While in pattern mode, the one of the colours can be altered like in normal drawing, in order to alter the other user must first select the background colour menu item. Note that if you try to flip diher state during intermediate actions, this effect will take place after the intermediate action is over, otherways it would cause problems to the complement drawing method used.

Apart from drawing, Hugs has many image processing capabilities. These effects are introduced by selecting an operation gadget (first two gadget groups). The first group launches a *type A* operation, the second a *type B*. Following is an explanation of each gadget's function.

Zoom

Launches a *get_position* action, positioning a rectangle reflecting the actual size of the zoom window over the drawing area. When done, the new position selected will be the one magnified. Zoom concept is thoroughly explained in the following section (*Operability*).

Cut

Launches a cut procedure to get a rectangular area and copies its contents to the clip buffer. This clip can be then pasted as many times as desired, using the next gadget.

Paste

Used to paste the contents of the clip buffer, no matter what source filled it. It launches a *get position* action, positioning a rectangle reflecting the clip size over the drawing area. There are various paste modes, activated by the special menu item *Paste mode*. Jam #2 completely overwrites the clip over the image, and inverse video does the same, inverting the paste colours. Jam #1 (the default), is like Jam #2, but is careful with colour 0 of the clip, which is transparent, ie. it does not affect the background. Complement mode is not all that useful, if you paste two times in the same position you will get back to the original image. The paste operation will repeat itself till manually aborted by selecting the same or another type A operation gadget.

In Huge there is no such thing as an eraser, but there is a smart way of going around this one. The user can cut a black (colour 0) portion of any size and paste it over the area to be cleared using Jam #2 paste mode, thus simulating a custom size eraser.

Fill

Launches a get point action, and flood fills the image with whatever colour and pattern currently selected, beginning with the point provided by user. Flood fills are sometimes time consuming, so Huge always enter the wait state until the operation is over. Like paste, fill autorepeats.

Mirrors & rotators

On the second row of type A operation gadgets there are two mirrors, left-right and up-down and two rotators, anti-clockwise and clockwise. Every operation expects a rectangular area to be input first, like in cutting, and then acts on it. The clip buffer is by no means affected. The rotated version will have common the upper left corner with the original. Note that when there is difference between vertical and horizontal screen resolution, the rotated version will be a bit stretched. Best results are achieved in full-HIRES mode or in full-LORES, when resolutions are same. All these operations are time consuming, so Huge enters the wait state till they are through.

All following gadgets are of type B

Font selector

Invokes the font selection requester, used to select the font that will be used in subsequent text writing.

Write text

The user is prompted to input the text to be displayed. This text is rendered in the clip buffer, using current font, text style and foreground colour. Immediately after that, a paste operation is initiated (paste gadget automatically selected), allowing user to position text on the drawing area.

Palette

Invokes the palette requester, used to modify existing palette.

Undo

Undoes last action. This action may be a flood fill, a paste or even a single dot. For memory conservation reasons there is no possibility of un-undo, once you undo you can't get back to what you originally had. When temporary lines are drawn (ie. action or drawing in progress), Huge prevents undo. After finishing the intermediate work the user should try the undo gadget again.

Workbench

Brings workbench screen in front, enabling multitasking operations. This gadget could be totally omitted if users remembered the standard Intuition keyboard shortcuts for this job: left Amiga-N brings workbench in front, and left Amiga-M sends workbench to the back, in any Amiga program.

Page #

Huge has two pages for drawing, the one behind of the other. In this way they don't have to be simultaneously present, increasing the available space for drawing. The second image is necessary for icons with alternate image, or simply as a backup area for comparisons. The user will soon realise that a big area available for drawing is not only useful for creating huge images, it can stand for a drawing workbench. This gadget brings the page behind in front. Like undo, Huge prevents page swapping during intermediate actions.

Operability

Huge is a true multitasking program. The user is absolutely free to choose the sequence of his/her actions. For example he can select fill, do a couple of fillings here and there, change colour and fill some more, load another palette, modify its colours a bit and resume filling. This is of course an extreme example, normally palette is determined before drawing, but it depicts the freedom user has with using Huge.

Keyboard equivalents

Huge is packed with keyboard equivalents for almost every gadget or menu operation. The mouse is a great input device, but it sometimes gets annoying moving down and up from draw to control screen to select another operation. For example it is very tiring having to move up and select background colour from the dither subitems of special menu, when compared to a right Amiga key press. Easy to remember keyboard shortcuts enable constant drawing without ever moving to the control screen.

Menu shortcuts are displayed next to corresponding items, user must press the right Amiga button simultaneously with the appropriate key to select an item, and all this from another screen than the one holding the menus (this is harder than it seems). The most frequently used menu commands have straightforward equivalents, even I admit that some are hard to remember, but I ran out of keys. This is the era for extremely intelligent people, only the sharpest brains will survive.

Keyboard shortcuts are available for every gadget, too. Keyboard events concerning gadgets are distinguished from those concerning menu items, because the right Amiga key is not pressed. When a key is recognised, the corresponding operation is initiated, just as if the gadget was pressed. TOGGLESELECT gadgets have their selection status flipped, but simple gadgets show no sign that they were selected. Following is a listing of gadget shortcuts:

a. Type A gadgets:

Zoom z
Cut c
Paste p
Fill f
L-R mirror..... l
U-D mirror d
ACLK rotate 9
CLK rotate..... 0

b. Type B gadgets:

Select font a
Write text t
Palette r (for Rainbow)
Undo u
Workbench w
Page # b (for Behind)

c. Tool gadgets:

Freehand 1
Freeline 2
Continuous lines.. 3
Circle..... 4
Ellipse..... 5
Rectangle 6

Another vital point, especially in type A operations, is exact coordinates input, and Huge has plenty features for easing this strife. The current position is displayed as a (x,y) coordinate in the coordinates window. The origin is the top left drawing area vertex. Coordinates display is user selectable via a toggleselect menu item in special menu (*Coords*). Shutting off coordinates will speed up things a bit, since whenever a new pointer position is detected it is displayed, but the gain is not that much, since the responsible routine is quite fast. My advice is to keep ~~coordinates~~ ON all the time.

Zoom concept

The most exact mean of specifying current position is to use zoom window, where each pixel is displayed magnified by a ratio depending on current screen resolution. This

resolution can be further customised using *Zoom resolution* subitem of the Zoom item on Special menu, in order to force variable magnification ratios. Initially a default resolution is selected, user can afterwards change it and it retains its value throughout the program run, until changed again. When screen changes mode, this resolution may have to be adjusted.

Normally the zoom window is updated each time an operation is over (ie. after fill, or paste is over), meaning that the intermediate lines won't be seen 'live' magnified on zoom window. By pressing right Amiga-u (ie. activating the Special toggleselct Zoom menu subitem *Live zoom*, u for update) the status changes; the zoom gets live, showing all intermediate lines and moving along with current position, trying always to keep the hot spot centred in it (if possible). While in 'live' zoom mode the right mouse button can be pressed to prevent zoom window moving along with current position, still it updates every time a new (intermediate) position is detected. Moreover, if the coordinates input happens over the zoom window, the zoom does not move along with current position, no matter if we are in trailer or static mode.

An example will help clearing up the mess. Say there is a get_size action in progress and you strive to get the exact rectangle size. Turn 'live' mode ON by pressing the RA-u combination and the zoom begins updating and moving along with the current position. Bring the pointer over the actual drawing area in order to roughly determine the desired size. Press the right button to prevent zoom moving around and move over to the zoom window to fine tune size. If you don't stop zoom from moving around, by the time you moved over the zoom window, its position will change, and that's not a desired thing. After you are done, you may want to switch OFF 'live' state by pressing the RA-u combination again. I admit that the zoom update routine is slow, and it gets worse as screen resolution and depth increases. This is actually the only drawback I know for this program that will be addressed in a future release.

The whole procedure is easily mastered, given a proper amount of experimenting. The 'live' and 'static' statuses are toggled with the RA-u combination and right mouse button respectively. Initially they are both OFF, and once you flip them ON, remain in the same state till flipped again.

While drawing, normally only the freehand tool immediately updates zoom, the rest tools update zoom when the drawing sequence is over. The RA-u combination alters this state while in drawing, too. The only problem is that when the current drawing tool is continuous lines, the 'trailer' state cannot be altered, since right mouse button clicks are filtered by this tool, signalling the beginning of a new line.

Finally, users should not forget the standard Intuition mouse shortcuts. Pressing either Amiga key simultaneously with a cursor key moves the pointer in the cursor direction. This is the most accurate method of moving exactly one pixel away from the current position. Left-Alt+left Amiga simulate left mouse button click and right-Alt+right Amiga simulate right mouse button click.

Input/Output

Huge Input/Output consists of object I/O and secondary I/O (palette and preferences). In order to load an object, user must first select *Object type* (and language for Images & Sprites) from the preferences menu. Next he must select *Open* from the project menu, and loader is invoked. After selecting the desired file, Huge opens and validates user selection. If everything is valid, Huge builds the object in the clip rastport and launches a paste operation, ie. user can select where to put the loaded object on the current page. The only exception are icons with alternate image highlighting. In this case, the secondary image is placed at the top left corner of the back page.

If Huge detects that the object just loaded was created in a different display, before launching a paste operation asks user if he/she desires to alter screen, in order to reflect the original settings. User can either allow or prevent display changes. If object is bigger than current display it will be properly clipped, it may also lose some colours if this screen has less colours than the one used when it was created and saved. Moreover, brushes are the only objects containing palette information, so Huge asks if it is to load the palette from file to screen colour registers.

In order to save an object, again object type must be specified (can be different than the one used for loading), and the *Save* item from project menu must be selected. Then a cut operation is initiated, filling the clip rastport with the portion of the drawing area desired to be saved. When the cut sequence is through, saver is invoked who prompts for a filename for the target object. After a valid filename is specified, the appropriate save routine for the current object is invoked, saving the clip contents. Note that for icons with image highlighting, the visible cut is saved as primary icon and the invisible, exactly behind the portion cut, is saved as selected icon.

For all objects except icons, user may want to save an icon along with the actual image data, by setting the toggleselect Special menu item *Save Icon* (default=no icon saved). This icon can be afterwards be used to start Huge from workbench by double clicking on it, via the default tool mechanism or extended selection. Huge automatically loads the selected object and the user can immediately begin editing the object. For more information see *Startup options* section following below.

Palettes follow a similar procedure, except for cutting and pasting, of course. Palette I/O is launched by selecting the desired *Palette* subitem of the Preferences menu.

Finally the *Save prefs* item of the Preferences menu can be used to save the current user preferences to a file. The default name is 'huge.preferences', but the user can specify a different one in order to keep many different configurations. See *Startup options* section to understand how the preferences file Huge loads at init time is determined. This preferences file contains current object, display (resolution & depth), zoom resolution and the palette.

Description of the Huge I/O system

The I/O system is extremely complicated internally. It is intelligent, meaning that knows enough to predict if an attempted action will be successful, before the OS, thus preventing actions that would launch a lousy OS requester. The idea is hard for the programmer, easy for the user. I am positive about the programmer part, I hope for the

user part. Huge maintains and updates internally, lists of every registered device, assigned directory and volume of the Amiga. It can tell if a volume is mounted or write protected, or if a drive is empty. Plenty of self-explaining messages inform user of what is going on.

At init time, Huge selects an appropriate lock (directory) to begin with (see Startup options section). While user is interacting with the I/O system, this lock will most probably change to other directories. Sometimes Huge has no lock at all, this loss caused by various reasons. For example, if you remove the disk Huge loaded from and try to use I/O, the lock evaporates. A more subtle loss is caused when trying to save. A lock may be perfectly suitable for loading, but if the disk is write protected and the saver is invoked, the lock goes down the drain. When Huge has no lock, no load/save can happen until a new lock is established. Huge will accept no filenames or simple directories, complaining about his lock. In order to establish a new lock, a full path has to be specified, ie. an input in the form:

device:[directory[/directory[...]]]

in a string gadget that accepts directories. Loader provides more ways around this one, since you can click on an existing, full device, or a mounted volume, providing easily the desired full path and thus new lock.

Before exiting, Huge restores the current directory to the one selected before it was started. At any time a lock exists, Huge has the current directory set to this lock, thus allowing file access using the plain name, stripped of paths.

Using requesters

Requesters are means of user input other than drawing. All open on the control screen, putting temporarily Huge in the wait state, till they are satisfied. Every requester, except for simple messages, provides at least two ways of exiting, an affirmative which accepts all changes introduced (OK), and a negative which is a safe way out when changes introduced are no longer desired (CANCEL). The two top left gadgets in every requester are for this reason. The leftmost is OK, the one on its right side is CANCEL. Input happens by clicking on gadgets, either simple, toggleselect, proportional or string gadgets, where user inputs strings or numbers. Wherever you see a hole a sting gadget is waiting for you. Following is a description of all requesters.

Messages

The most frequently encountered requester, especially for new users. A string scrolls notifying user of what is going on. Most of them are simple, meaning that user is just notified and just accepts the message. A few need an answer from the user, positive signalled by clicking OK, negative by clicking CANCEL. For example when user attempts to quit without saving changes, a message will inform him/her of the situation and ask for confirmation before proceeding.

Simple input

This one is launched whenever user must enter a string or a number. Such requesters are launched before writing text, and when altering icon characteristics (default tool & stack size)

Icon tool types

This is a complex string input requester, activated by selecting the *Tool types* item on Icon menu. It functions just like the workbench *info* counterpart. Initially, current icon's tool types are displayed and may be edited. Use the arrows to see the rest entries, and '+' '-' to add or delete an entry, accordingly.

Zoom resolution

Activated by selecting the *Zoom resolution* subitem on the Special menu. Through this one, user can fine tune the zoom resolution, ie. how many times magnified will a real pixel be seen on the zoom window. This is controlled by a single value since horizontal and vertical magnifications are same. Minimum allowable value is 2 (1 means displaying the original imagery, it is no magnification) and maximum depends on the current display.

The new magnification ratio can be inserted either by clicking on the arrow gadgets, or by directly entering the numeric value in the little string gadget. Intermediate steps have immediate effect on the draw screen, so user can fine tune magnification.

Notice that due to horizontal or vertical resolution differences of the current display, a pixel may not be rectangular, a fact that is also reflected in it's magnified representation. Huge makes sure that the zoom window is always rectangular, despite any resolution differences. Moreover, Huge prevents resolution changes in the middle of a zoom positioning operation.

Screen mode

Activated by selecting the *Screen* item of the Preferences menu. Here the desired display characteristics are specified. Two toggleselect gadgets adjust the horizontal and vertical resolution (HIRES and LACE), if they are selected, you will get the high resolution in the corresponding direction.

Next is screen depth. The deeper the screen, the more colours available (and less memory...). The formula is:

$$\text{colours} = 2^{\text{depth}}$$

Maximum depth allowed is 5 (32 colours), and if this is a HIRES screen, maximum depth is limited to 4 bitplanes, because Amiga chips can't produce a 32 colour display in high resolution (not enough DMA time say guys who know...). You can change depth by either hitting an arrow gadget or by directly writing the desired number in the small string (longint) gadget.

Palette

Activated by selecting the palette gadget (type B), enables modification of current palette. This is the only requester that accepts input from the draw window besides from its gadgets.

In order to alter a colour, you select it from the draw screen, as in normal drawing. The actual RGB values that make up this colour are displayed both as numbers and proportional indications. You alter a colour by modifying one or more of these RGB values, either by directly entering a number in the corresponding colour string gadget, or by using the proportional counterpart. Each colour is a mixture of the three basic colours (Red, Green & Blue) in various proportions, varying from 0 to 15 parts, enabling a total of $16 \times 16 \times 16 = 4096$ different colours.

Some fancy palette operations are included as well. *Swap*, *Copy* and *Shade*, resemble the type A operation gadgets. They are toggleselect, they need another colour selected by user, so they can act on two colours: the freshly selected, and the one that was already selected when this type of gadget was selected. After a valid second colour is selected (eg. it can't be the same colour), they do what their name implies. User is free, at anytime before selecting that second colour, to select another operation, even the same, thus deactivating it. *WBench*, *User* and *Reset*, give an easy way of selecting standard palettes, the standard workbench, the user preferences and Huge's standard palette accordingly. Finally the undo gadget undoes last change introduced.

note: Huge's standard palette is the one loaded from the preferences file.

Font selector

Invoked by clicking on the font select gadget. All the currently available fonts are present, alphabetically sorted by name (ROM fonts first). Each font name encloses many point sizes of the same typeface. The available sizes of the currently selected typeface are listed on the right side of the names, sorted in ascending size. A sample of the currently selected font is displayed on the left side of the requester.

User can try different point sizes of the current typeface or a different typeface. You can browse through available fonts using the proportional gadgets on the right of each listing. Names and heights can be specified either by clicking on the desired item, or by directly entering name or size in one of the available string gadgets. If a new typeface is selected, Huge automatically opens and displays its first point size.

Huge won't open a diskfont if the volume containing FONTS: directory is not mounted, except if it was previously opened.

Loader

The most complicated of requesters (actually a window), is invoked whenever user tries to open something. The type of object being opened is displayed on the window title.

If a lock exists, loader starts reading current directory's contents and displays them at the same time it is watching for user input. The main goal here is to select an existing filename for loading, maybe by changing initial directory, too.

The most of the gadgets are means of easy directory selection. There are four device gadgets to access the root directory of the corresponding device, one (':') moving quickly to the root directory of the currently active drive, one for moving up one directory level ('/'=parent), a string gadget for directory input, either absolute or relative, and finally two see-and-click ways. Loader displays all registered volumes first, the subdirectories next and the files last. Each type is distinguished by it's unique colour. The last two methods of directory change is by either clicking on a volume name or subdirectory. In the path string gadget, the full path of the current directory is displayed.

Initially, and every time a new directory is set, the file input string gadget is automatically activated, so user can immediately type an already known filename. Otherwise, the desired file can be selected by browsing through available file entries and clicking.

Saver

Called each time a filename must be provided for saving. This one checks if there is a valid lock, and if so fills the sole string gadget with a default name. The path gadget provides the current full path. Remember that in order to save the disk must be write enabled. A full path may be entered if desired in the form:

`[[device:][directory/[directory/[.../]]]]filename`

Full path is inspected for validity, and if approved Saver changes current directory to the one freshly specified and displays just the filename stripped from paths. This one also checks if a file with the same name already exists in the current directory and if yes, prompts user for an overwrite confirmation.

Using menus

Menus are used to launch more complex actions, or to set user defined options. Huge automatically disables menu items that are not applicable at a certain moment or configuration. Following is an explanation of those menu items left unexplained.

Project menu

New clears the current project (visible page only), and performs various initialisations (eg. icon characteristics except type & hilite fall back on default values). The clip buffer remains intact. Any possible drawing or operation in progress is aborted. *About* shows some credits, and *Quit* provides the normal exit.

Watch the *New* item's side effect on icon characteristics. *New* does not simply clear the imagery, clears icon characteristics, too. This fact can have a catastrophic impact on project icons, since the default tool is essential for their use. It is a common practice when using the icon editor aspect of Huge to modify existing icons' imagery, to load the original icon, clear the display and draw the replacement. If this clearing is done through choosing *New*, all the characteristics of the original icon will be junked. In order to avoid this side effect, users must clear the imagery with the custom eraser trick noted above. You may think that this item's performance is awkward, but the icon initialisations are essential. Suppose you are constructing a series of icons for your needs, and each one has different characteristics (stack size, tool types, etc.). If you don't use *New* to clear the characteristics of the previous icon, the same will be saved for the next one, which may not be desired. Instead of having to clear each characteristic separately, there is the *New* item that does the job for you. Normally, when you open an existing icon, all its characteristics are loaded, overwriting previous user settings. This item has no side effects on the rest object types.

Preferences menu

Here you select current *Object* type and *Language* using the corresponding submenus. Remember the sprite and icon display limitations I was talking about. If the display is not the appropriate for the current object, the screen will change, or the object change won't take place. The *Workbench* submenu opens or closes (if possible) workbench screen on user request. Closing workbench is a good idea for 512K Amigas.

In order to convert an object to a type that poses display limitations, users should use the clip buffer who retains its contents during display changes. If you don't use the clip buffer, unless the current screen mode is the same required for the target object (eg. Icon or Sprite), while trying to alter object type Huge will ask for a display change thus losing the original imagery. So before attempting to alter object type, cut the desired imagery to the clip rastport (it is already there if you just have loaded the object). Of course you won't be able to use the same number of colours, and maybe resolution will be different than the original object type. This is a factor that you should account for from the beginning, eg. if you design a Brush using a paint package other than Huge planning to use it as an Icon, you shouldn't draw it in anything but a standard icon display.

Icon menu

This menu is active only when the object selected is icon. Using the various items of this menu, user has complete control over all icon characteristics, just beware that this extensive power can do more harm than good in the hands of the unaware. If you are not sure about any item's effect on icon performance, don't mess around with it, otherwise icon failure may result. It is safe enough to load an existing icon, alter its imagery and save back using the same name, maybe altering highlighting to *Image* for simple icons (any highlighting method change introduced is quite safe).

Huge can be used either for altering current icons or for constructing new icons, eg. for your new program. This program is used to customise the icon aspect (visible image & workbench properties), it does not perform any actions in the direction of the filesystem part of icons. This means that when you save a drawer icon, Huge does not check if a directory with the same name exists on the target disk, neither creates one. In the same fashion, it does not check if there is a tool or project with the same name, or forces 'disk.info' names for disk icons. Just remember that whatever you do must have a filesystem counterpart.

Special menu

Here are various special effects. In order to select *Background colour* from the *Dither* submenu, the pattern mode must be ON. When you select this item nothing happens, just the next colour you select will be treated as background colour. *Paper* item is similar, ie. when you select it nothing happens, but the next colour you select will be used for redrawing draw screen's paper. Normally Huge uses the last colour for paper, but sometimes this colour ruins the display.

Finally there are two items concerning fonts. The available fonts are determined at init time, but user may want to use another set of fonts by setting another directory as the FONTS: one. *Read fonts* can be used to read the new font lists. Besides, if at init time Huge realised that the volume containing FONTS: directory wasn't mounted, it proceeds using just the ROM fonts, this item comes then to rescue. Just remember to insert the right volume this time. The *Style* submenu modifies text characteristics. When a new font is opened by the font selector, this submenu defaults to *Normal* style. User may afterwards add any or all of the available text styles before writing. By manually selecting normal style, all other styles are switched OFF.

Requirements

Huge, like any other program involving screens is memory hungry. Closing Workbench is a good idea, or switching off any external drives. For the icon stuff to work, *icon.library* must be in the current LBS: directory, otherwise the program will run as usual, but with all icon aspects disabled. In a similar fashion, *diskfont.library* should be in LBS: directory, otherwise Huge will not service any font operations, ie. no text.

Finally, I observed that when *FastFonts* is installed, the scrolling messages get jerky. If you get jerky scrollings, check your startup-sequence for *FF*. If this is the case, run *FF -n* from CLI and the scroll will be fine.

Memory conservation

Huge uses the least possible memory for its display rasters but can do no magic tricks. Either you have the memory or you don't. Following, some aspects of Huge are explained that may provide insight for better memory management.

Disk based libraries

Hide away from Huge LIBS: directory, if icon & diskfont libraries aren't already open (icon.library is surely open if you started Huge from Workbench). This way you save about 10K, sacrificing font & icon use. This 10K can prove invaluable guys. If you want to edit icons of course you must let Huge find icon.library. Anyway icons never cause memory troubles, since they have just 4 colours. You start getting memory trouble as the screen gets deeper.

If you eventually decide to use fonts, beware that when you open a font, it will remain resident until dooms day. There is no OS blessed procedure for removing a loaded diskfont (*RemFont()* is a fake, it just removes the font from the system list, instead of sending it where it came from, as it should). So you shouldn't open curiously all available fonts if you have a 512K Amiga. Have a test ride, opening all fonts, find out which one you like most, reboot the machine and then select only this favourite one. Another way is to remove FONTS: directory from Huge's sight, leaving you with the ROM fonts (Topaz 8 & 9), which are quite nice.

Raster memory

There are no ways around this one. Either your Amiga has the sufficient memory, or it doesn't. If Huge while attempting to set new screen realises that there is not enough memory available, notifies user and attempts to restore the original screen. In the case where this isn't possible (I can't imagine why), Huge exits gracefully. This type of memory includes actual display memory and two object-sized rastports, the one holding the page behind and the other being the undo buffer.

I/O and clip rastport

Keep in mind that you don't just need enough memory for the desired display, there must be left a good amount of memory for operations and I/O. Imagine yourself having just finished a superb full screen bitmap and receiving a 'Not enough memory for saving' message.

The clip rastport is shared among various different Huge's operations. Initially there is none allocated. Each time some Huge subsystem needs to access the clip buffer, junks the old one (if any) and allocates a new whose size is just sufficient. This means that if you cut a screen portion and then write some text, clip buffer will contain the text, not the original cut. Similar is the case while saving & loading. If you initiate a save sequence, the clip buffer will contain the portion you selected to save, and loader puts the loaded object in the clip as well. The clip buffer survives screen changes. The contents of this buffer remain intact until modified by an operation, till then it can be pasted as many times as desired, *in any page*. Note that Jam #1 and Complement paste modes pose a memory overhead at paste time, needed for a single plane mask. This means that it is possible to receive a 'Not enough memory for pasting' message when using these modes.

All this means that for normal operations Huge uses a variable amount of memory, depending on the current clip size, which can't be greater than an object sized rastport.

Besides the memory used for the clip buffer, while loading or saving **Huge** needs to find a good amount of memory for I/O buffers. This amount depends on the current object type. Images are quite economical on buffer use as they allocate a buffer to hold just one plane of the image. Brushes are the most memory hungry object since their planes are interleaved and the I/O buffer is as great as the actual image. Besides no compression is performed, even if all paint programs save big brushes using a compacting routine. **Huge** will load of course compacted brushes. Bobs are the least memory hungry objects and should be used as a last chance object if every other object causes a 'Not enough memory for loading/saving' message. In fact bobs use no extra I/O buffer at all, the same buffer that holds the imagery (clip rastport) is used for I/O buffer as well. At least you will be able to save your creation and visit a friend whose Amiga has expansion memory to do the final job. Loading poses memory overhead any time the loaded object is bigger than maximum object size for the current display, since it must be fully loaded and then transplanted to a maximum size clip buffer.

Desperate guys can minimally configure their Amiga before starting **Huge**, by interrupting the startup-sequence before it loads any NEWCON: or RAM: handlers. If this doesn't work, it's time for that expansion memory.

Note that the feature of **Huge** of putting the loaded object in the clip rastport and enabling its positioning over the drawing area is a powerful clipping concept, simulating a disk based clip rastport with variable clip names and no memory overhead, the only drawback being the time required for disk access. This can be compensated by using the RAM disk for temporary clips.

Startup options

Huge can be started by either CLI or Workbench. In both ways there are options for specifying the preferences file to be used, for overriding default object type and specifying an object file Huge is to load immediately after it starts running. The initial directory is also determined.

CLI start

The huge command line can take the form:

huge [-options] [[path:]filename]

options available:

-p <prefs file>
-o <object type>
 i <c|a> ... image (C or assembly)
 s <c|a> ... sprite (C or assembly)
 b brush
 B bob
 I icon

It is not necessary to provide any arguments. If you do, note that case is important, and no spaces exist between the letter that specifies option type and the option parameters. In the rare case that you might want to load an object whose name begins with either '-p' or '-o', use capitals (eg. '-P') to escape the conflict, since AmigaDOS is not case sensitive (I want to see that user).

The prefs file will be used to read display, zoom resolution, palette and default object type. If no particular preferences filename is specified, Huge will attempt to open a file named 'huge.preferences' from the initial directory. If such a file does not exist, defaults will be used. The object option is a way to override default object type specified by the preferences file. This way you can have a single preferences file to use a display and its colours with various objects. The initial directory in CLI starts is the current one. If the filename specified contains a valid path, this will be the initial directory. It is a good idea to give a full path for the prefs file since initial directory can change by the method just described, thus making impossible to locate prefs file.

sample command line:

huge -pprefs:sprite.prefs -osc ram:sleep.pointer

Workbench start

Huge can be started by various methods. Normally by double clicking on its icon, by default tool mechanism if an icon saved along with a Huge object is double clicked, and finally by extended selection. If Huge is started alone, the initial directory will be that including Huge. The desired preferences file is located by examining a PREFS=<file> tool type of Huge's icon. If no such entry exists, again 'huge.preferences' is used.

If Huge is started along with a project, either by default tool mechanism or extended selection, the initial directory will be that of the project's. Huge projects have automatically set a FILETYPE=<object_type> tool type entry, which is used to determine object type. If no such entry exists, icon is assumed. Remember that Huge does not save an icon for icons. The easiest way to edit an icon of any type (disk, drawer, etc.) is to click (once) on the Huge icon, hold the SHIFT key, and double click on the desired icon. Extended selection is a must for icons, since probably no icon will have Huge as the default tool, besides other icon types may be desired to be edited apart from project icons.

Huge puts no PREFS=<prefs_file> tool type entry for the icons it saves along with actual object data, but the user can manually set such an entry, in order to provide the desired preferences file for the specific object. This entry, overrides any possible PREFS=<...> entry on Huge icon tool types.

Bibliography

Listed below are volumes I have extensively used while developing this program. Users wishing to program the Amiga simply can't do without them.

- [1] Amiga ROM Kernel reference manual: Libraries & Devices (Addison-Wesley, 1989)
- [2] Amiga ROM Kernel reference manual: Includes & Autodocs (Addison-Wesley, 1990)
- [3] Amiga C for advanced programmers (Abacus, 1990)
- [4] Amiga Machine language (Abacus, 1990)
- [5] AmigaDOS inside & out (Abacus 1989)
- [6] A500/A2000 User's manual (Commodore, 1987)
- [7] A500/A2000 AmigaBASIC reference manual (Commodore, 1985)

A final word

Huge has been thoroughly checked, and I have never seen a *Guru* caused by it. Every possible error situation is supposedly anticipated and overcome gracefully. Nevertheless, I can't guarantee flawless operation. Every #7.0 major upgrade may contain some subtle errors that programmer failed to anticipate. Any bug reports and suggestions are appreciated.

Using Huge properly will improve the looks of the new programs to come, and this is all for the better. Just keep peoples' eyes and soul satisfied.

UPRIGHT!

Nick & Panos Bozines,
5 Gregoriou E' St.,
GR 54 248 THESSALONIKI,
GREECE

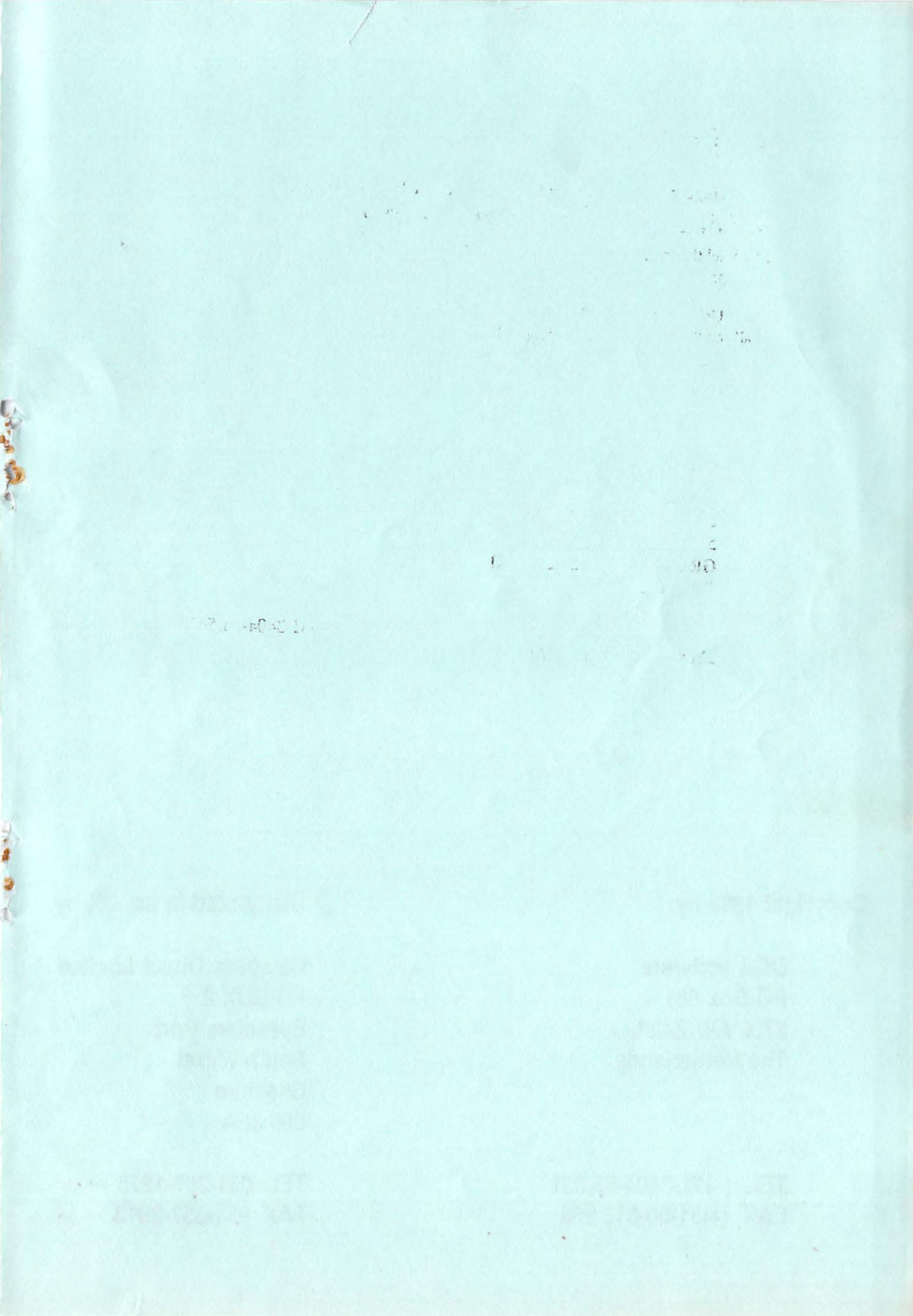
TEL (+)30-31-319.940
FAX (+)30-31-907.011

UGA software

P.O. Box 881,
3700 AW Zeist,
The Netherlands

TEL (+)31-3404-53.531
FAX (+)31-30-313.968

Manual compiled on 26 September, 1991



Copyright 1991 by:

UGA software
PO Box 881
3700 AW Zeist
The Netherlands

TEL (+)31-3404-53.531
FAX (+)31-30-313.968

Distributed in the UK by

Database Direct Limited
PO BOX 2
Ellesmere Port
South Wirral
Cheshire
L65 3EA

TEL 051-357-1275
FAX 051-357-2813